REMARKS

Status of the Claims

Claims 1, 5, 6, 9, 10, 15, 19, 20, 23, and 27-29 are currently presented in the Application, and claims 1, 10, and 15 are independent claims. In this Amendment, claims 1, 9, 10, and 15 have been amended, claims 2-4, 7, 8, 11-14, 16-18, 21, 22, and 24-26 have been canceled, and new claims 27-29 have been added. Applicant is not conceding in this Application that the canceled claims are not patentable over the art cited by the Examiner, as the present claim amendments and cancellations are only for facilitating expeditious prosecution of the Application. Applicant respectfully reserves the right to pursue these and other claims in one or more continuation and/or divisional patent applications.

In particular, Applicant has amended independent claim 1 to include limitations similar to those previously found in dependent claims 2-4, 7, and 8. Similarly, Applicant has amended independent claims 10 and 15 to include limitations similar to those previously found in claims 11-14, and 16-18, 21, and 22, respectively. New claims 27-29 have been added to include information handling system claims similar to method claims 5, 6, and 9. No new matter has been added as a result of these amendments.

Examiner Interview

Applicant wishes to thank Examiner Black for the courtesy extended to Applicant's attorney during a telephone interview on August 23, 2007. During the interview, Applicant's attorney discussed the Kundu reference with regard to independent claim 1 (see full argument below). In particular, Applicant's attorney pointed out that Kundu is concerned with database checkpoints, and does not appear to be concerned with nullifying variables. Examiner Black indicated that she would perform a new search upon receiving Applicant's formal response. No agreement was reached regarding the claims during the interview.

Claim Rejections – Alleged Obviousness Under 35 U.S.C. § 103

Claims 1-26 stand rejected under 35 U.S.C. § 103(a) as being obvious over Sayag, U.S. Patent No. 6,898,602 (hereinafter Sayag), in view of Kundu et al., U.S.

Publication No. 2004/0210577 (hereinafter Kundu), and further in view of Kolawa et al., U.S. Patent No. 5,842,019 (hereinafter Kolawa). Applicant respectfully traverses the rejections under 35 U.S.C. § 103.

A. There Is No Motivation To Combine Sayag, Kundu, And Kolawa

Regarding the three references cited in the Office Action, Sayag discloses the "use of garbage collection in order to determine the exact amount of memory that is consumed by a running application at any point of its execution" (see Sayag, Abstract). Kolawa discloses "dynamically detecting leaked memory space in a computer program" (see Kolawa, Abstract). Both Sayag and Kolawa deal with the general technology area of runtime memory space. Sayag is concerned with determining the amount of memory space used by a runtime application, while Kolawa is concerned with detecting memory space leaks during runtime. Kundu, on the other hand, is concerned with a completely different technology area. Kundu discloses "[t]echniques for making light-weight checkpoints in logs of streams of transactions" (see Kundu, Abstract). Kundu uses the redo logs that are typically used in database systems to log transactions, and purports to increase the usefulness of these redo logs for purposes such as data mining and replication of transactions. As discussed in detail in Kundu, a logical redo log is made from a physical redo log. Light-weight checkpoints are made and used in the logical redo log, and these light-weight checkpoints are used in data mining and replication (Kundu, paragraph [0041]).

The Final Office Action states that a skilled artisan would be motivated to combine these three references, and in particular to combine Kundu with Sayag in order to allocate more space within memory to store data as well as to maintain memory management in a system (see Final Office Action, page 15). The Final Office Action also notes that Kundu teaches making checkpoints in a logical redo log by inserting checkpoint logical change records at the proper locations in the logical redo log. The Final Office Action then asserts that "a redo log is interpreted to be a set of files that record all changes made to an [sic] database and a checkpoint is interpreted to be an identified snapshot of a database or a point at which the transactions against the database have been frozen, ***which is equivalent to nullifying the identified last used***

variable" (Final Office Action, page 15, emphasis added). Applicant respectfully disagrees. Taking a snapshot of a database and/or freezing transactions is not equivalent to nullifying an identified last used variable. The purpose of checkpointing and freezing transactions is to have a record of the state of a database at a particular point in time. Checkpointing and freezing transactions do not involve nullifying variables.

Kundu is not concerned with, and indeed does not even address, the issue of memory space as used by runtime applications. Kundu is in a completely different technology area from Sayag and Kolawa. Kundu is concerned with database and DBMS technology, whereas Sayag and Kolawa deal with runtime memory space. Other than the fact that all three references have to do with the very general area of computer science, there is simply no justification to combine Kundu with either Sayag or Kolawa, and a skilled artisan would not be motivated to combine these three references.

B. The Cited References Do Not Teach Or Suggest All The Limitations Of Applicant's Claims

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). If an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). Manual of Patent Examining Procedure § 2143.03. Applicant respectfully submits that none of the cited references, either alone or in combination, teaches or suggests all the elements of Applicant's claims.

Using independent claim 1 as an exemplary claim, Applicant teaches and claims the following:

- reading, by a just in time compiler, one or more variables included in one or more activation records included in the middleware computer program, wherein the

variables reference one or more objects, the objects stored in a garbage collected heap stored in a computer memory;

- identifying, by the just in time compiler, a program statement in the middleware computer program where a selected variable is last used;
- inserting, by the just in time compiler, a nullification statement after the identified program statement, the nullification statement adapted to nullify the selected variable;
- writing the activation records, the identified program statement, and the nullification statement to a resulting code file for the middleware computer program;
- executing the middleware computer program, wherein the executing further comprises:
    - executing the nullification statement to nullify the selected variable; and
    - after nullifying the selected variable, executing a garbage collection program, wherein executing the garbage collection program comprises identifying one of the objects that was previously referenced by the selected variable, and reclaiming the memory occupied by the identified object.

As an initial matter, Applicant notes that claim 1 has been amended to clarify that the reading, identifying, and inserting steps are performed by a just in time compiler. The Final Office Action asserts that Sayag discloses the limitation of reading variables included in activation records, but correctly notes that Sayag does not teach or suggest Applicant's "identifying" and "inserting" limitations (see Final Office Action, page 4). The Final Office Action then cites Kundu as disclosing identifying a program statement in a

computer program where a variable is last used, and inserting a nullification statement after the identified program statement (see Final Office Action, pages 4-5). Applicant respectfully disagrees. As discussed above, Kundu discloses "[t]echniques for making light-weight checkpoints in logs of streams of transactions" (see Kundu, Abstract). Kundu uses the redo logs that are typically used in database systems to log transactions, and purports to increase the usefulness of these redo logs for purposes such as data mining and replication of transactions. As discussed in detail in Kundu, a logical redo log is made from a physical redo log. Light-weight checkpoints are made and used in the logical redo log, and these light-weight checkpoints are used in data mining and replication (Kundu, paragraph [0041]).

The Office Action cites Kundu at paragraph [0051] as teaching the "identifying" step of Applicant's independent claims. The cited section of Kundu discusses "the information needed to make a logical redo log from a physical redo log" (Kundu, paragraph [0050]). This information includes various fields, such as session#, client#, server#, session_name, etc. The Office Action asserts that Kundu's "checkpoint_scn" is somehow equivalent to "identifying a program statement in the middleware computer program where a selected variable is last used." Applicant respectfully disagrees. Kundu's checkpoint_scn is the system change number, SCN, of the most recently-made checkpoint in the physical redo log. As known to those skilled in the art, a checkpoint is a saved state of a program and its data, which can be used to restart the program at the point at which the checkpoint occurred. As explained in Kundu, "locations in physical redo logs are identified by system change numbers" (Kundu, paragraph [0051]). These system change numbers are used to identify locations within a physical redo log. Applicant fails to see how a system change number can be said to read on Applicant's step of "identifying a program statement in the middleware computer program where a selected variable is last used." Kundu does not appear to be concerned with determining where, within a program, a variable is last used. The physical redo logs in Kundu are concerned with taking a snapshot of a database program at a particular point in time. The variables used within that time frame may or may not continue to be used

after the checkpoint is taken. Kundu simply does not address determining where a variable is last used.

The Final Office Action reiterates its point that "a redo log is interpreted to be a set of files that record all changes made to an [sic] database and a checkpoint is interpreted to be an identified snapshot of a database or a point at which the transactions against the database have been frozen, ***which is equivalent to nullifying the identified last used variable***" (Final Office Action, page 25, emphasis added). As discussed above, Applicant respectfully disagrees. Taking a snapshot of a database and/or freezing transactions is not equivalent to nullifying an identified last used variable. Checkpointing and freezing transactions do not involve nullifying variables.

The cited section of Kundu at paragraph [0066] discusses producing a logical redo log. Checkpoints are made in the logical redo log by inserting checkpoint logical change records, LCRs, at the proper locations in the logical redo log. The Final Office Action asserts that inserting a logical change record into a logical redo log is equivalent to "inserting a nullification statement after the identified program statement," but then admits that Kundu does not disclose "the nullification statement adapted to nullify the selected variable" (see Final Office Action, page 25). Applicant respectfully reminds the Examiner that MPEP § 2143.03 states that all the limitations of a claim must be considered. As stated in MPEP § 2143.03:

2143.03 All Claim Limitations Must Be Taught or Suggested

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). If an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988).

Applicant clearly claims that the nullification statement that is inserted after the identified program statement is meant "to nullify the selected variable." Kundu does not teach or suggest a nullification statement that nullifies a variable. Kundu discusses inserting logical change records into a logical redo log. A logical change record is not

the same as a nullification statement. Even assuming, for the sake of argument (and Applicant does not agree that this is the case) that a logical change record could somehow be said to be equivalent to a nullification statement, the logical change records disclosed by Kundo do not nullify a variable. Kundu does not teach or suggest nullifying a variable, and Kundu certainly does not teach or suggest nullifying a variable by inserting a nullification statement after a program statement in which the variable is last used. Further, the Final Office Action admits that Kundu's logical change records do not nullify variables (see Final Office Action, page 25). Applicant specifically teaches and claims a nullification statement that nullifies a selected variable, and yet the Final Office Action admits that neither Kundu nor Sayag teaches this aspect of Applicant's claim.

The Office Action then uses yet another reference, i.e. Kolawa, to disclose "the nullification statement adapted to nullify the selected variable" (see Final Office Action, page 26) However, the cited section of Kolawa at col. 5, lines 40-61, discusses a routine for performing leak searching. Allocated memory space is tracked using reference counting. A reference count is assigned to a pointer, and if the reference count becomes zero, it indicates that a memory leak may exist. Setting a reference count to zero does not nullify any variables, nor does it nullify the pointer. As clearly shown in Figure 7 of Kolawa, when the reference count equals zero, it indicates that there may be a memory leak, and thus it is time to do a memory search in order to determine if there is a memory leak. Using a reference count to determine that a potential memory leak exists is simply not the same as "inserting a nullification statement after the identified program statement, the nullification statement adapted to nullify the selected variable," as taught and claimed by Applicant. Kolawa does not appear to be concerned with nullifying variables.

Applicant has amended claim 1 to specifically claim writing the activation records, the identified program statement, and the nullification statement to a resulting code file for the middleware computer program and then executing the middleware computer program. During execution of the middleware computer program, the nullification statement is executed in order to nullify the selected variable (i.e. after the selected

variable is last used). A garbage collection program is then executed and reclaims memory occupied by an object that was previously referenced by the selected variable. None of the cited references, either alone in combination, teach or suggest these elements of claim 1. As discussed above, none of the cited references teach or suggest nullifying a variable after it is last used in a program in order to reclaim the memory occupied by objects previously referenced by the variable.

Because none of the cited references, either alone or in combination, teach or suggest all the elements of independent claim 1, Applicant respectfully submits that independent claim 1 is patentable over the cited references. Independent claims 10 and 15 include limitations similar to those in independent claim 1, and are therefore patentable for at least the reasons discussed above with regard to claim 1. Therefore, Applicant respectfully submits that independent claims 1, 10, and 15, and the claims which depend from them, are patentable over Sayag in view of Kundu and Kolawa.

Conclusion

As a result of the foregoing, it is asserted by Applicant that the remaining claims in the Application are in condition for allowance, and Applicant respectfully requests an early allowance of such claims.

Applicant respectfully request that the Examiner contact the Applicant's attorney listed below if the Examiner believes that such a discussion would be helpful in resolving any remaining questions or issues related to this Application.

Respectfully submitted,

By /Leslie A. Van Leeuwen, Reg. No. 42,196/
Leslie A. Van Leeuwen, Reg. No. 42,196
Van Leeuwen & Van Leeuwen
Attorneys for Applicant
Telephone: (512) 301-6738
Facsimile: (512) 301-6742